

A small health clinic with three doctors operates in a village. All clients of the clinic have their details stored in the clinic's database. Patients that visit the clinic during the day are given a priority rating (1–3) and are seated in a waiting room to wait for the next available doctor. When it is their turn, the patients are taken from the waiting room to have a consultation with their assigned doctor, who makes a diagnosis, provides treatment and writes a prescription.

The clinic's system is coded in Java. There are many objects in this system and some of them are listed below.

Object	Description
Doctor	A licensed professional who treats patients in the clinic.
Patient	A sick person who requires a consultation with a doctor.
WaitingRoom	A place where patients wait for their consultations.
Consultation	A dated meeting between a doctor and a patient which results in a diagnosis, treatment and a prescription for medication.
Treatment	A dated record of all actions and medication prescribed to treat the patient's diagnosed condition.

The three objects `Patient`, `WaitingRoom` and `Treatment` have been defined in the following UML diagrams:

<b>Patient</b> Integer id String name Integer priority String doctor  setId (Integer id) setName (String name) setPriority (Integer priority) setDoctor (String doctor) Integer getID() String getName() Integer getPriority() String getDoctor() String toString()	<b>Treatment</b> String date Integer patientId String doctor String actions String medication  setDate (String date) setPatientId (Integer id) setDoctor (String doctor) setActions (String actions) setMedication (String medication) String getDate() Integer getPatientID() String getDoctor() String getActions() String getMedication() String toString()
<b>WaitingRoom</b> Patient[10]patients  add(Patient newPatient) void callNextPatient() Integer findNextPatientIndex() remove(Integer n)	

The Patient and WaitingRoom objects are implemented as follows:

```
public class Patient
{
    private int id;
    private String name;
    private int priority;
    private String doctor;

    public Patient(int i, String n, int p)
    {
        id = i;
        name = n;
        priority = p;
        doctor = null;
    }
    public void setId(int i) { id = i; }
    public void setName(String n) { name = n; }
    public void setPriority(int p) { priority = p; }
    public void setDoctor(String d) { doctor = d; }
    public int getId() { return id; }
    public String getName() { return name; }
    public int getPriority() { return priority; }
    public String getDoctor() { return doctor; }
    public String toString() { return id+" "+name+" "+priority+" "+doctor; }
}

public class WaitingRoom
{
    private Patient[] patients = new Patients[10];

    // uses default constructor

    public void add(Patient newPatient)
    // adds the new patient in the next empty array location
    {
        int i = 0;
        while ((patients[i] != null) && (i < 10))
        {
            i=i+1;
        }
        if (i==10) { System.out.println("No more space in the waiting room."); }
        else { patients[i] = newPatient; }
    }
}
```

```

public void callNextPatient()
// finds the next patient, outputs their details
// and removes the patient from the array
{
    int index = 0;
    if (patients[0]==null)
    {
        System.out.println("The waiting room is empty.");
    }
    else
    {
        index = findNextPatientIndex();
        remove(index);
    }
}

private int findNextPatientIndex()
// returns the index of the first patient with the
// highest priority in the array patients
{
    int max = 0;
    //... code missing ...
    return max;
}

private void remove(int n)
// outputs the data of the patient instance at array index n
// and removes that patient by shifting all remaining patients
// by one index towards the front of the array
{
    //... code missing ...
}
}

```

14. (a) Define the term *constructor*, using an example from the code on pages 14 and 15. [2]
- (b) Describe **one** additional field that might have been included in the `Patient` class. Include a data type and sample data in your answer. [2]
- (c) Describe the relationship between the `Patient` object and the `WaitingRoom` object. [2]

Consider the `WaitingRoom` class as presented on pages 14 and 15.

- (d) Construct the missing lines of code in the `findNextPatientIndex()` method to return the index of the first patient with the highest priority in the `patients` array. **Note:** the highest possible priority is 3. [3]
- (e) Construct the `remove(int n)` method which outputs the data of the patient object at index `n` and then removes that patient object by moving all remaining patient objects one index towards the front of the `patients` array. You may assume that `n` is a valid index between 0 and 9, and that an instance of `Patient` exists at that index. [6]
15. (a) In relation to the `Patient` class, outline **one** advantage of encapsulation. [2]
- (b) In relation to the `Treatment` object, discuss **one** ethical consideration when designing software that stores patients and their illnesses. [4]

The clinic would like to start storing details in a `Doctor` object, including full name, telephone number and whether the doctor is present or not. For example:

name:	Dr Henriëtte Mănescu-Rața
phone:	0734511122
present:	true

- (c) Design the `Doctor` object using a UML diagram. [3]
- (d) In relation to the `Doctor` object, outline the need for extended character sets as used by modern programming languages. [3]

14. (a) Award **[1]** for a definition, such as:  
A (special) method (with the same name as the class) that instantiates an object of that class;

Award **[1]** for the example:

```
public Patient(String i, String n, int p)
```

[2]

- (b) Award **[1]** for a field and its data type.

For example:

```
String phoneNumber;
```

[2]

- (c) Award **[1]** for any indication of aggregation.  
Award an additional **[1]** for a full description.

Example answer:

The WaitingRoom object stores up to 10 instances of the Patient object;

The WaitingRoom has a Patient;

Accept a correctly labelled diagram.

[2]

- (d) Award marks as follows up to **[3 max]**.  
Award **[2]** for correct loop, award **[1]** for loop with one condition.  
Award **[1]** for correct test.  
Award **[1]** for assigning max.

Example answer:

```
int i=1;
while ((i<10) && (patients[i]!=null))
{
    if (patients[i].priority > patients[max].priority)
    {
        max = i;
    }
    i = i+1;
}
```

[3]

- (e) Award marks as follows up to **[6 max]**.  
Award **[1]** for including the correct signature.  
Award **[1]** for correct output of patient details.  
Award **[1]** for any loop.  
Award **[1]** for correct loop.  
Award **[1]** for correct assignment to shift a patient instance.  
Award **[1]** for assigning null.

Example answer:

```
public void remove(int n)
{
    System.out.println(patients[n].toString());
    for (int i=n; i<9; i=i+1)
    {
        patients[i] = patients[i+1];
    }
    patients[9] = null;
}
```

[6]

15. (a) Award **[1]** for a suitable definition, for example:  
Encapsulation means having private variables;  
Variables not directly accessible from outside the class;  
Methods and variables are all included in the class definition;

Award **[1]** for relating an advantage to the *Patient* class, such as:  
So that patient data cannot be accessed/modified accidentally;  
So that patient data is more secure;

[2]

- (b) Award **[1]** for identifying an ethical issue and **[1]** for an elaboration.  
Award **[1]** for relating this issue to the *Treatment* class and **[1]** for an elaboration.

Example answer:

Sensitive information should be separated from identity data;  
In order to avoid information misuse;  
The consultation object does not include the patient's name (only an ID);  
So that a sensitive diagnosis (like AIDS) is not directly linked to a name;

[4]

- (c) Award marks as follows up to **[3 max]**.  
Award **[1]** for including *String name* with correct getter and setter methods.  
Award **[1]** for including *String phone* with correct getter and setter methods.  
Award **[1]** for including *Boolean present* with correct getter and setter methods;

Example answer:

Doctor
String name
String phone
<b>Boolean</b> present
setName(String name)
setPhone(String phone)
setPresent(Boolean present)
String getName()
String getPhone()
<b>Boolean</b> getPresent()
String toString() //optional

[3]

- (d) Award **[1]** for identifying an appropriate need for extended character sets and an additional **[1]** for an elaboration of this need. Award **[1]** for relating to the *Doctor* object.

Example answer:

Different languages use different characters, which are not included in the basic 8-bit ASCII character set;  
Extended character sets (like Unicode) include all possible characters from all languages;  
For example, the doctor's name could not be spelled correctly in ASCII;

[3]